Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On deciding synchronizability for asynchronously communicating systems ☆

Samik Basu [a,*], Tevfik Bultan [b]

[a] Iowa State University, United States
[b] University of California at Santa Barbara, United States

ARTICLE INFO

Article history:
Received 6 August 2015
Received in revised form 4 August 2016
Accepted 26 September 2016
Available online xxxx
Communicated by P. Aziz Abdulla

Keywords:
Asynchronous systems
Message-passing systems
Synchronizability
Verification

ABSTRACT

Asynchronously communicating systems involve peers or entities that communicate by exchanging messages via buffers. In general, the size of such buffers is not known apriori, i.e., they are considered to be unbounded. As a result, models of asynchronously communicating systems typically exhibit infinite state spaces and it is well-known that reachability and boundedness problems for such models are undecidable. This, in turn, makes automatic verification of asynchronous systems undecidable as well. We discuss a particular class of asynchronous systems over peers for which the interaction behaviors do not change when the peers are made to communicate synchronously. Such systems are referred to as *Synchronizable*. Automatic verification of synchronizable systems is decidable as the verification of the system can be performed using its synchronous counterpart. Recently, we have proved that checking whether or not a system is synchronizable is decidable. In this paper, we consider different types of asynchronous communication, where the type is described in terms of the nature of buffering and the number of buffers, and discuss how/if synchronizability is decidable for each type. The new results subsume the existing ones and present a comprehensive synchronizability study of asynchronous systems.

1. Introduction

With the increasing use of software systems in distributed settings, dependability of distributed systems has remained one of the crucial problems in computing. A distributed system with many components coordinates their executions in order to achieve a desired objective. One emerging paradigm to realize such coordination is based on message-based interaction. In this setting, the components (we refer to them as *peers*) interact by exchanging (sending and receiving) messages and coordinate their activities/execution [1–6].

Due to the distributed nature of the system, the exchange of messages does not occur in a lock-step fashion. That is, a message sent by a peer is not immediately consumed by the receiving peer owing to delays in the communication network. In other words, the sender and the receiver are not always in sync—the system resulting from the communication is called *asynchronous system*. The model of such asynchrony involves buffers—messages sent by the sender are stored in the

☆ This work was supported by the National Science Foundation, under grant CCF1116836.
* Corresponding author.
E-mail addresses: sbasu@iastate.edu (S. Basu), bultan@cs.ucsb.edu (T. Bultan).

http://dx.doi.org/10.1016/j.tcs.2016.09.023
0304-3975/

Please cite this article in press as: S. Basu, T. Bultan, On deciding synchronizability for asynchronously communicating systems, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2016.09.023

buffers and messages consumed by the receivers are removed from the buffers. The capacity of the buffer is assumed to be unbounded to capture any and all possible delays that can incur between the sending of the message and its consumption. The unboundedness in the capacity results in models of asynchronous system that exhibit infinite state-space and, in fact, such systems can simulate Turing Machines [7]. This renders their automatic verification, that relies on state-space exploration, undecidable, in general.

As a result, a number of techniques have been developed to identify different subclasses of asynchronous systems for which automatic verification is decidable and tractable (e.g., [8,9]). One such class considered in [10,11] is referred to as the *synchronizable systems*. An asynchronous system over peers is said to be synchronizable if and only if any behavior exhibited by the asynchronous system with unbounded buffers can be exhibited by the same peers when communicating synchronously (i.e., with no buffers). In synchronous systems, the state-space is finitely bounded (by the product of the number of states of each peers) and, therefore, automatic verification of synchronous systems can be performed efficiently. In other words, automatic verification of an asynchronous system, that is synchronizable, can be performed by verifying its synchronous variant.

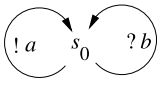
The question remains: "*Is synchronizability checking decidable?*". In [12,13], we proved that synchronizability checking is decidable for asynchronous systems where (a) each receiver peer has a buffer, and (b) each buffer acts as a queue (FIFO).

We consider the behavior of the systems in terms of sequences of sends, which can be viewed as the observable behavior of a system. The consumption of messages (receives) is typically viewed as local to the receiving peers and is not considered observable. The choice for focusing on the sequences of send actions stems from the fact in distributed systems such as Web services, the consumption of messages (receives) is typically viewed as local to the receiving services and is considered unobservable. Furthermore, the desired behaviors of the services are often described in terms of the messages being exchanged (sent) by the participating peers/services [14]. Similar, specifications describe the desired interactions in Singularity OS communication contracts [15] and UBF(B) communication contracts in distributed Erlang programs [16].

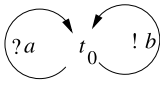
We have proved that an asynchronous system (denoted by $\mathcal{I}$) over a given set of peers described as finite-state machines is synchronizable if and only if the sequences of sends in the corresponding synchronous system (denoted by $\mathcal{I}_0$), where peers communicate synchronously, are identical to that in the 1-bounded asynchronous system (denoted by $\mathcal{I}_1$), where peers communicated asynchronously via buffers of capacity 1. Given that both $\mathcal{I}_0$ and $\mathcal{I}_1$ exhibit behavior with finite-state space, verifying whether or not they have the same set of send-sequences can be performed automatically, which, in turn, makes synchronizability checking decidable.

As a simple example, consider three peers as follows:

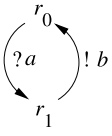
- $\mathcal{P}_1$ has one state s_0 with two loops: one over send action a and other over receive action b.



- $\mathcal{P}_2$ has one state t_0 with two loops: one over receive action a and the other over send action b.



- $\mathcal{P}_3$ has two states r_0 and r_1. It has a transition from r_0 to r_1 on receive action a and a transition from r_1 to r_0 on send action b.



Assume that the start states of each peer are subscripted by 0. Consider that the synchronous communication between $\mathcal{P}_1$ and $\mathcal{P}_2$ results in $\mathcal{I}_0$. In $\mathcal{I}_0$, every time $\mathcal{P}_1$ sends message a by executing $!a$, $\mathcal{P}_2$ is ready to consume it synchronously (in lock-step) by executing $?a$; similarly, every time $\mathcal{P}_2$ sends message b, it can be immediately consumed by $\mathcal{P}_1$. As a result of synchronous interactions, the sequence of sends will be any ordering of a's and b's. The same sequences are also present in their 1-bounded asynchronous system $\mathcal{I}_1$—where the sent messages are not necessarily immediately consumed; instead the messages are stored in a buffer of size 1. Therefore, based on [12], the asynchronous system resulting from $\mathcal{P}_1$ and $\mathcal{P}_2$ is synchronizable.

On the other hand, the synchronous communication between $\mathcal{P}_1$ and $\mathcal{P}_3$ results in sending of a, which is immediately consumed by $\mathcal{P}_3$ and is followed by sending of b by $\mathcal{P}_3$, which, in turn, is consumed synchronously by $\mathcal{P}_1$. The resultant sequence of sends is $ababab\ldots$. However, the 1-bounded asynchronous system involving the interaction between the same

Please cite this article in press as: S. Basu, T. Bultan, On deciding synchronizability for asynchronously communicating systems, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2016.09.023